Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнил:

Студент группы P3210

Цыпандин Н. П.

Преподаватели:

Косяков М.С.

Тараканов Д. С.

Санкт-Петербург

2022

Задача №1 (E) «Коровы в стойла»

```cpp
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

vector<int> positions;

bool can_be_placed(int potential_answer, int count);

int main() {
    int n, k;
    cin >> n >> k;

    positions.resize(n);
    for (size_t i = 0; i < n; ++i)
        cin >> positions[i];

    int left = 0, right = positions.back() - positions.front(), mid;
    while (left < right) {
        mid = (left + right) / 2;
        if (can_be_placed(mid, k))
            left = mid + 1;
        else
            right = mid;
    }

    if (can_be_placed(left, k))
        cout << left << endl;
    else
        cout << left - 1 << endl;

    return 0;
}

bool can_be_placed(int potential_answer, int count) {
    int left = positions.front();
    count--;
    for (size_t i = 1; i < positions.size(); ++i) {
        if ((positions[i] - left) >= potential_answer) {
            left = positions[i];
            count--;
        }
        if (count == 0)
            return true;
    }
    return false;
}
```

Пояснение к примененному алгоритму:

Решим задачу с помощью бинарного поиска по ответу. Потенциальный ответ будем проверять на соответствие проходясь по всем позициям стойл. Если оказывается, что расстояние между предыдущей поставленной коровой и текущей >= потенциального ответа, то ставим корову. Так пытаемся поставить всех k коров.

Асимптотика: O(N * logN)

Задача №2 (F) «Число»

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

bool strips_comp(string s1, string s2) {
    return s1 + s2 > s2 + s1;
}

int main() {
    string str;
    vector<string> strips;

    while (cin >> str)
        strips.push_back(str);

    sort(strips.begin(), strips.end(), strips_comp);

    for (auto &strip: strips)
        cout << strip;
    return 0;
}
```

Пояснение к примененному алгоритму:

Главное написать правильный компаратор для последующей сортировки строк. Будем сортировать в лексикографическом порядке по убыванию. Полученные кусочки объединим воедино.

Асимптотика: O(N * logN)

P.S. Т.к. количество входных данных невелико, прошла мы и простая сортировка за N^2 (например пузырьковая, вставками или выбором)

Задача №3 (G) «Кошмар в замке»

```
#include <iostream>
#include <string>
#include <algorithm>
#include <vector>

using namespace std;

typedef struct weight_and_char {
    size_t weight;
    char c;
    size_t cnt;
} element;

bool comp(element e1, element e2) {
    return e1.weight > e2.weight;
}

int main() {
    string str;
    cin >> str;

    /**
     * Посчитаем сколько каждой из букв
     */
    const int alphabet_sz = 26;
    vector<size_t> char_cnt(alphabet_sz, 0);
    for (char i: str)
        char_cnt[i - 'a']++;


    /**
     * Отсортируем все буквы по весам, независимо от количества
     */
    vector<element> weights(alphabet_sz);
    for (int i = 0; i < alphabet_sz; ++i) {
        size_t weight;
        cin >> weight;
        weights[i] = (element)
                {
                        .weight = weight,
                        .c = char('a' + i),
                        .cnt = char_cnt[i]
                };
    }
    sort(weights.begin(), weights.end(), comp);

    /**
     * Если букв хотя бы 2, то припишем их слева и справа
     * Так мы получим максимальную пользу от конкретной буквы с весом
     * Если букв меньше 2, то они нам не важны, засунем их позже в центр
     */
    vector<char> ans(str.length());
    size_t left = 0, right = ans.size() - 1;
    for (size_t i = 0; i < alphabet_sz; ++i) {
        if (weights[i].cnt < 2)
            continue;
```

```
            ans[left] = weights[i].c;
            ans[right] = weights[i].c;
            --right;
            ++left;
        }

        /**
         * Припишем оставшиеся незначимые буквы
         */
        for (size_t i = 0; i < alphabet_sz; ++i) {
            if (weights[i].cnt == 1) {
                ans[left] = weights[i].c;
                ++left;
            } else if (weights[i].cnt > 2) {
                for (size_t j = 0; j < weights[i].cnt - 2; ++j) {
                    ans[left] = weights[i].c;
                    ++left;
                }
            }
        }

        for (char an: ans)
            cout << an;

        return 0;
}
```

Пояснение к примененному алгоритму:

Заведем структуру, хранящую информацию о конкретной букве, её весе и количества вхождений. Всего будет 26 элементов, отсортируем все это дело по убыванию веса. Далее, будем поступат жадно, если количество вхождений не меньше 2, тогда расставим по одной букве слева и справа поддерживая левый и правый указатели. Далее припишем оставшиеся незначимые буквы в центр.

Асимптотика: O(N)

(т.к. сортируем всего 26 элементов, нам нужно всего лишь проходиться по строке что бы получить от неё информацию о количестве вхождений каждой из букв)

Задача №4 (H) «Магазин»

```cpp
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

bool comp(int a, int b) {
    return a > b;
}

int main() {
    int n, k;
    cin >> n >> k;

    vector<int> costs(n);
    for (size_t i = 0; i < n; ++i) {
        cin >> costs[i];
    }

    sort(costs.begin(), costs.end(), comp);

    int ans = 0;
    for (size_t i = 0; i < n; ++i) {
        if ((i + 1) % k != 0)
            ans += costs[i];
    }

    cout << ans << endl;
    return 0;
}
```

Пояснение к примененному алгоритму:

Поступим жадно, отсортируем все товары по убыванию цены, и далее будем считать общую стоимость товаров пропуская каждый k-ый товар.

Асимптотика: O(N * logN)